MTI #1792
Docket No. 18153.0054

CONFIGURATION FUSES FOR SETTING PWM OPTIONS

FIELD OF THE INVENTION:

The present invention relates in general to output signal generation and, more particularly, to a pulse width modulation (PWM) generator that includes configuration bits for placing PWM output signals into tri-state, active high or active low modes when the PWM module is inactive or when individual PWM outputs are not enabled.

BACKGROUND OF THE INVENTION:

Processors, including microprocessors, digital signal processors and microcontrollers, operate by running software programs that are embodied in one or more series of program instructions stored in a memory. The processors run the software by fetching the program instructions from the series of program instructions, decoding the program instructions and executing them. In addition to program instructions, data is also stored in memory that is accessible by the processor. Generally, the program instructions process data by accessing data in memory, modifying the data and storing the modified data into memory.

Processors may be programmed to perform a wide variety of functions in software. In some cases, however, dedicated hardware may be included in a processor that significantly eases the processing load needed to perform certain functions. This allows the use of lower performance processor for these functions, which lowers the cost of the processor. One type of dedicated hardware that may advantageously be included in a processor is power control hardware. Power control hardware provides the capability to control circuitry and devices that use significant amounts of power. For example, power control hardware may be used to control motors, power supplies, etc.

One common mode of operation of power control hardware is pulse width modulation (PWM). In PWM, the power level is controlled by controlling the duty cycle of a signal that has only two states - active and inactive. The signal is then connected to output transistors and a load, such as a motor, to yield the equivalent of a continuously varying voltage and current.

When PWM hardware is included in a processor, external switching devices, such as transistors, must be used in order to handle significant amounts of power. These switching devices have less than perfect switching characteristics, especially when connected to devices such as motors. Problems arise with conventional PWM hardware, which has been included in current processors, in dealing with the less than perfect switching characteristics of connected switching devices.

When processors including PWM hardware drive externals devices, care must be taken to ensure that the external devices are driven to proper values at all times. This is done to avoid indeterminate states being applied to the external devices, the application of which would cause high power dissipation in and possibly damage to external devices.

In order to overcome these problems, conventional processors incorporating PWM hardware have incorporated tri-state output buffers. The tri-state output buffers cause the PWM outputs to "float" or not produce an output voltage in a tri-state mode. The tri-state mode is entered whenever the PWM module is inactive. Conventional systems also incorporate helping devices, such as resistors, which tie the PWM output pins to a termination voltage (generally either power or ground) according to the requirements of the system. The helping devices are active while the PWM output pins are tri-stated to avoid indeterminate states being applied to the external devices. The helping devices may also be weak enough that they remain active while

the PWM outputs are active. In this scenario, the PWM outputs overcome the helping devices in applying output signals to the external devices in an active PWM mode.

While the conventional approach of incorporating external helping devices, such as resistors, works, it requires additional hardware, additional space on printed circuit boards to accommodate the devices and may increase the cost to implement. Moreover, conventional approaches may not permit simple system re-configuration.

Accordingly, there is a need for a new system and method for ensuring that determinate states are applied via PWM outputs under inactive conditions of the PWM module to external devices. There is a further need for a flexible approach to solving the problem that does not require resistors or devices external to the processor.

SUMMARY OF THE INVENTION:

According to the present invention, configuration bits are provided that configure PWM outputs of a processor incorporating a PWM module. The configuration bits cause the PWM module to put the PWM outputs into tri-state, active high or active low modes when the PWM module is inactive or when individual PWM outputs are not enabled. The configuration bits are stored in non-volatile memory and perform the configuration after power-up of the processor and after a reset when the PWM module is generally in an inactive state.

According to an embodiment of the invention, a pulse width modulator for a processor is provided that has pulse width modulation outputs that are configured to operate based on configuration bits. The pulse width modulator includes output control logic, configuration bits and an output transistor pair. The configuration bits selectively configure output control logic. The output transistor pair has upper and lower transistors that are respectively coupled to

opposing output voltage levels. When the output transistor pair is not enabled for PWM data output, the configuration bits cause the output control logic to configure the output pair to operate in one of a tri-state, active high or active low mode.

The configuration bits include a tri-state control bit that configures the output pair to operate in one of a tri-state or an active mode.

The configuration bits may further include a high device set bit that configures the output pair to operate in one of an active high or an active low mode when the tri-state control bit configures the output pair to operate in an active mode. The high device set bit may configure output pairs that output a signal to an external device in a high-side driver arrangement, such as transistor 302A.

The configuration bits may further include a low device set bit that configures the output pair to operate in one of an active high or an active low mode when the tri-state control bit configures the output pair to operate in an active mode. The low device set bit may configure output pairs that output a signal to an external device in a low-side driver arrangement, such as transistor 302B.

BRIEF DESCRIPTION OF THE FIGURES:

Fig. 1 depicts a functional block diagram of an embodiment of a processor chip within which the present invention may find application.

Fig. 2 depicts a functional block diagram of a pulse width modulation (PWM) module for use in a processor, such as that shown in Fig. 1.

Fig. 3 depicts an exemplary external circuit, which may be driven by complementary PWM signals generated by the circuit shown in Fig. 2.

Fig. 4 depicts an illustration of a prior art embodiment for preventing indeterminate voltages from being applied to external devices when PWM outputs are not enabled.

Fig. 5 depicts an embodiment of the present invention for using configuration bits stored in non-volatile memory to control PWM output signal generation when PWM data outputs are not enabled.

DETAILED DESCRIPTION:

According to the present invention, configuration bits are provided that configure PWM outputs of a processor incorporating a PWM module. The configuration bits cause the PWM module to put the PWM outputs into tri-state, active high or active low modes when the PWM module is inactive or when individual PWM outputs are not enabled. The configuration bits are stored in non-volatile memory and perform the configuration after power-up of the processor and after a reset when the PWM module is generally in an inactive state.

Fig. 1 depicts a functional block diagram of an embodiment of a processor chip within which the present invention may find application. Referring to Fig. 1, a processor 100 is coupled to external devices/systems 140. The processor 100 may be any type of processor including, for example, a digital signal processor (DSP), a microprocessor, a microcontroller, or combinations thereof. The external devices 140 may be any type of systems or devices including input/output devices such as keyboards, displays, speakers, microphones, memory, or other systems which may or may not include processors. Moreover, the processor 100 and the external devices 140 may together comprise a stand alone system.

The processor 100 includes a program memory 105, an instruction fetch/decode unit 110, instruction execution units 115, data memory and registers 120, peripherals 125, data I/O 130,

and a program counter and loop control unit 135. The bus 150, which may include one or more common buses, communicates data between the units as shown.

The program memory 105 stores software embodied in program instructions for execution by the processor 100. The program memory 105 may comprise any type of nonvolatile memory such as a read only memory (ROM), a programmable read only memory (PROM), an electrically programmable or an electrically programmable and erasable read only memory (EPROM or EEPROM) or flash memory. In addition, the program memory 105 may be supplemented with external nonvolatile memory 145 as shown to increase the complexity of software available to the processor 100. Alternatively, the program memory may be volatile memory, which receives program instructions from, for example, an external non-volatile memory 145. When the program memory 105 is nonvolatile memory, the program memory may be programmed at the time of manufacturing the processor 100 or prior to or during implementation of the processor 100 within a system. In the latter scenario, the processor 100 may be programmed through a process called in-circuit serial programming.

The instruction fetch/decode unit 110 is coupled to the program memory 105, the instruction execution units 115, and the data memory 120. Coupled to the program memory 105 and the bus 150 is the program counter and loop control unit 135. The instruction fetch/decode unit 110 fetches the instructions from the program memory 105 specified by the address value contained in the program counter 135. The instruction fetch/decode unit 110 then decodes the fetched instructions and sends the decoded instructions to the appropriate execution unit 115. The instruction fetch/decode unit 110 may also send operand information including addresses of data to the data memory 120 and to functional elements that access the registers.

The program counter and loop control unit 135 includes a program counter register (not shown) which stores an address of the next instruction to be fetched. During normal instruction processing, the program counter register may be incremented to cause sequential instructions to be fetched. Alternatively, the program counter value may be altered by loading a new value into it via the bus 150. The new value may be derived based on decoding and executing a flow control instruction such as, for example, a branch instruction. In addition, the loop control portion of the program counter and loop control unit 135 may be used to provide repeat instruction processing and repeat loop control as further described below.

The instruction execution units 115 receive the decoded instructions from the instruction fetch/decode unit 110 and thereafter execute the decoded instructions. As part of this process, the execution units may retrieve one or two operands via the bus 150 and store the result into a register or memory location within the data memory 120. The execution units may include an arithmetic logic unit (ALU) such as those typically found in a microcontroller. The execution units may also include a digital signal processing engine, a floating point processor, an integer processor, or any other convenient execution unit.

The data memory and registers 120 are volatile memory and are used to store data used and generated by the execution units. The data memory 120 and program memory 105 are preferably separate memories for storing data and program instructions respectively. This format is a known generally as a Harvard architecture. It is noted, however, that according to the present invention, the architecture may be a Von-Neuman architecture or a modified Harvard architecture, which permits the use of some program space for data space. A dotted line is shown, for example, connecting the program memory 105 to the bus 150. This path may include

logic for aligning data reads from program space such as, for example, during table reads from program space to data memory 120.

A plurality of peripherals 125 on the processor may be coupled to the bus 125. The peripherals may include pulse width modulation (PWM) module 160 and other peripherals 165, such as analog to digital converters, timers, bus interfaces and protocols such as, for example, the controller area network (CAN) protocol or the Universal Serial Bus (USB) protocol and other peripherals. The peripherals exchange data over the bus 150 with the other units. The PWM module 160 is capable of generating multiple, synchronized pulse width modulated (PWM) outputs. The PWM module 160 may be advantageously applied to a variety of power and motion control applications, such as control of Three-Phase AC Induction Motors, Switched Reluctance (SR) Motors, Brushless DC (BLDC) Motors, and Uninterruptable Power Supplies (UPSs).

The data I/O unit 130 may include transceivers and other logic for interfacing with the external devices/systems 140. The data I/O unit 130 may further include functionality to permit in circuit serial programming of the Program memory through the data I/O unit 130.

Fig. 2 depicts a functional block diagram of a pulse width modulation (PWM) module 160, for use in a processor 100, such as that shown in Fig. 1. PWM module 160 includes control registers 202, timebase 204, special event logic 206, at least one pulse width modulation generator, such as PWM generators 208A, 208B, 208C, and 208D, for each PWM generator, a deadtime generator, such as deadtime generators 210A, 210B, 210C, and 210D, and output driver circuitry 212. Bus 150 is communicates data among units of processor 100 and elements of PWM module 160. In particular, bus 150 communicates data with control registers 202,

timebase 204, special event logic 206, and the at least one PWM generator, such as PWM generators 208A, 208B, 208C, and 208D.

Control registers 202 store values that are modifiable in software and provide the capability to control and configure the operation of the elements of PWM module 160. Control registers 202 may include a plurality of control registers, each control register including a plurality of bits. Each control register may be read, under software control, to determine the configuration and operational state of elements of PWM module 160. Likewise, each control register may be written, under software control, to set the configuration and control the operation of element of PWM module 160. For example, bits in control registers 202 may provide the capability to enable or disable the generation of PWM signals by PWM module 160. Bits in control register 202 may provide the capability to set the polarity and timing of signals output by PWM module 160, such as the frequency, duty cycle, and dead time of such signals. Bits in control registers 202 may provide the capability to enable, disable, and configure special event triggering, fault triggering, override operation, and other functions of PWM module 160.

Timebase 204 generates timing signals that are used by other elements of PWM module 160, such as special event logic 206 and the PWM generators 206A-D. Timebase 204 may include registers, counters, comparators, and other circuitry that operate with a timing clock signal to provide the capability to generate timing signals having programmable parameters. For example, timebase 204 may provide the capability to program parameters such as: the count direction of included counters, the resolution and prescaling of the timing clock used to generate the timebase signals, the mode of operation of timebase interrupts, postscaling of timebase signals, and the mode of operations of the timebase, such as continuous, free running, single shot, etc.

Special event logic 206 generates trigger signals that provide the capability to synchronize operations external to PWM module 160 with the operation of PWM module 160. For example, in an embodiment of processor 100 in which other peripherals 165 includes one or more analog to digital (A/D) converters, the operation of such A/D converters may be synchronized to the operation of PWM module 160 using the trigger signals generated by special event logic 206. Special event logic 206 uses signals generated by timebase 204 to generate trigger signals that are synchronized with selected points in the period of the PWM signals generated by PWM module 160.

Each PWM generator generates a PWM signal, which is input to a deadtime generator. Each PWM generator, such as PWM generator 208A, may include a duty cycle register, such as duty cycle register 214, a comparator, such as comparator 216, and associated circuitry. Duty cycle register 214 stores a value that controls the duty cycle of the PWM signals. The duty cycle of a PWM signal is the fraction of each complete PWM cycle that the signal is in the active state. Duty cycle register 214 typically includes a buffer register, which is accessible by software, and a comparison register, which stores the actual compare value used in each PWM cycle. The value in the comparison register is compared by comparator 216, to a value generated by timebase 204. The status of this comparison controls the signals output from comparator 216, which, in turn, control whether the PWM signal is in the active or inactive state.

The output from each comparator, such as comparator 216, is input to a deadtime generator, such as deadtime generator 210A. Deadtime generator 210A may pass through the signal from comparator 216 without alteration, or deadtime generator 210A may alter the signal. Deadtime generator 210A may generate a set of complementary PWM signals based on the signal from comparator 216. Complementary signals are signals that are arranged so that when

one signal is active, the other signal is inactive. When the active signal becomes inactive, the inactive signal becomes active, and so on. Deadtime generator 210 also inserts deadtime into the complementary signals. Deadtime is a period during which neither complementary signal is active.

The complementary PWM signals generated by each deadtime generator, such as deadtime generator 210A, is input to output driver circuitry 212, which includes circuitry of sufficient capacity to drive circuitry external to processor 100. The drive signals are supplied to external circuitry via processor pins, such as pins 218A and 218B.

Fault inputs 220A and 220B provide the capability to drive one or more of the PWM outputs to a defined state. Such a function is useful if a fault occurs in the external circuitry that is controlled by the PWM outputs of processor 100. The function of the fault inputs is performed directly in hardware, so that fault events can be managed quickly. Examples of faults that may occur include failure of an external switching device, such as a transistor, short circuit of external circuitry or devices, such as a motor, overcurrent detected in external circuitry or devices, a fault in the power supply, etc. Fault pin priority logic 222 provides the capability to prioritize the function of the fault inputs if more than one input becomes active. The signals output from fault pin priority logic 222 are input to the deadtime generators 210A - 210D. The deadtime generators also include fault override logic that overrides the function of the deadtime generator in response to a fault signal from fault pin priority logic 222, if so configured.

Included in control registers 202 are registers that control the configuration and function of PWM module 160 in response to activation of one or more fault inputs. In particular, the registers provide the capability to define whether a particular pair of PWM outputs associated with a deadtime generator, such as PWM outputs 218A and 218B and deadtime generator 210A,

are controlled by the fault inputs. If enabled, the override logic in the deadtime generator will respond to a fault output signal 224 from fault pin priority logic 222 and perform a defined action.

Control registers 202 store values that define the state of each PWM output in response to a fault signal input to each fault input. Each PWM output can be defined to be driven inactive or active in response to the fault signal input to each fault input. For example, PWM output 218A may be defined to be driven inactive in response to a fault signal on fault input 220A and may be defined to be driven active in response to a fault signal on fault input 220B. If a PWM output pair associated with one deadtime generator is in the complementary output mode and both PWM outputs are defined to be driven to the active state in response to a fault signal input to a fault input, both PWM outputs would be driven active, which is not desirable. In this situation, the override logic in the deadtime generator will give priority to one PWM output, drive that PWM output active, and drive the other PWM output inactive.

Fault pin priority logic 222 provides prioritization among the fault inputs. If more than one fault input has been defined to control a particular PWM output and at least two such fault inputs become active concurrently, fault pin priority logic 222 selects one of the fault inputs to be given priority. The PWM outputs are driven to the fault states defined for the fault input that has been given priority, and the other fault inputs are ignored. Fault priority logic 222 generates a fault output signal 224 that indicates the selected fault input. Fault output signal 224 in input to the deadtime generators, such as deadtime generator 210A, which drives its associated PWM outputs to the fault state defined for the selected fault input.

Each of the fault inputs has two modes of operation:

Latched Mode: When the fault input is driven active, the PWM outputs will remain in the defined fault states until the fault input is driven inactive and the fault condition is cleared in software. The PWM outputs will be enabled for normal, non-fault operation once the fault condition is cleared in software.

Cycle-by-Cycle Mode: When the fault input is driven active, the PWM outputs will remain in the defined fault states until the fault input is driven inactive. When the fault input is driven inactive, the PWM outputs will return to normal, no-fault operation at the beginning of the next PWM period.

The mode of operation of each fault input is defined in registers included in control registers 202.

Each fault input may also be controlled directly by software. Processor 100 can be configured so that software can directly drive the active or inactive levels of each fault input.

An example of an external circuit that may be driven by complementary PWM signals is shown in Fig. 3. In this example, three transistor pairs, a first pair including transistors 302A and 302B, a second pair including transistors 304A and 304B, and a third pair including transistors 306A and 306B, are connected to complementary PWM outputs of processor 100, either directly or via appropriate additional circuitry. For example, the signal on pin 218A, shown in Fig. 2, may be connected to input 308A of transistor 302A and the complementary signal on pin 218B, also shown in Fig. 2, may be connected to input 308B of transistor 302B. One of skill in the art would recognize that pins 218A and 218B would typically be connected to inputs 308A and 308B, respectively, via appropriate, and well-known, circuitry. Other complementary PWM outputs from processor 100 may similarly be connected to inputs to other transistor pairs.

The output from each transistor pair is formed at a connection between the transistors in the pair. In the example shown, which uses transistors that are MOSFETs, the output of each transistor pair is formed at the connection between the source of the upper transistor and the drain of the lower transistor. For example, output 310A is formed at the connection of the source of transistor 302A and the drain of transistor 302B. The outputs of the transistor pairs are connected to windings of motor 312 and supply the power that drives motor 312.

In the example of Fig. 3, MOSFET transistors and a three-phase AC induction motor are illustrated. One of skill in the art would, of course, recognize that other types of transistors and other types of motors could be used as well as those illustrated. For example, transistors, such as bipolar transistors, insulated-gate bipolar transistors, and other well-known types of transistors, or motors, such as switched reluctance (SR) motors, or brushless DC (BLDC) motors could be used instead of those illustrated, with well-known modifications to the circuitry. Likewise, as is well-known, the PWM signals could be used to control other applications, such as switching power supplies, etc.

Fig. 4 depicts an illustration of a prior art embodiment for preventing indeterminate voltages from being applied to external devices. Referring to Fig. 4, a processor 400 incorporates PWM logic 405. The PWM logic 405 does not incorporate configuration bits according to the present invention for configuring PWM outputs. Rather, the PWM logic 405 produces PWM outputs 407 and 408 that are tri-stated when the PWM logic 405 is not active and produces PWM data when active.

The PWM output 407 is coupled to the gate of an external transistor 410, via appropriate and well known circuitry, that is an upper transistor connected to the power supply. The PWM output 408 is coupled to the gate of an external transistor 415, via appropriate and well known

circuitry, that is a lower transistor connected to the ground voltage. The transistors are coupled together and generate an output signal to drive a load such as the coil 420 as shown and as discussed above.

When the PWM logic is inactive, such as during and after a power up sequence and after a reset of the processor 400, the PWM outputs 407 and 408 are placed in a tri-state mode. For this reason, these outputs would "float" to indeterminate values causing power dissipation and possibly high current through the devices 410, 415 and 420. To counteract this problem, conventionally resistors 425 and 430 are coupled between the outputs 407 and 407 and termination voltages 435 and 440. The resistors 425 and 430 are generally small enough to allow charging the transistors 410 and 415 to known states in a reasonably short amount of time to avoid damage. However, the resistors are not large enough to overpower the PWM output signals when they are active.

The termination voltage 435 may be different from the termination voltage 440. This is because the upper and lower devices 410 and 415 respectively may require different polarity signals to place them in the "OFF" state. In general, the termination voltages 435 and 440 are chosen to place the transistors 410 and 415 into the OFF state when the PWM outputs 407 and 408 are tri-stated.

Fig. 5 depicts an embodiment of the present invention for using configuration bits stored in non-volatile memory to control PWM output signal generation to avoid the resistor scheme of the prior art if the user desires. Referring to Fig. 5, a non-volatile memory 500 is provided on processor 100 embodiments of the present invention or as part of the PWM module 160 according to the present invention. The non-volatile memory may be read only memory (ROM), electronically programmable read only memory (EPROM), electronically erasable

programmable read only memory (EEPROM), flash memory or any other type of non-volatile memory. The non-volatile memory 500 includes configuration bits as shown. The configuration bits include a tri-state control bit 505, a high device set bit 510 and a low device set bit 515. The configuration bits (the operation of each of which is explained below) are coupled to output control logic 520.

The output control logic 520 is coupled to output transistor pairs that each drive a PWM output signal. Two output transistor pairs for driving PWM outputs coupled to the upper and lower transistors of external devices are shown which may be representative, for example, of output signals 218A and 218B shown in Fig. 2.

The output transistor pair 522 drives a PWM output signal that is coupled to an upper external transistor that drives an external load such as transistor 302A shown in Fig. 3. The output transistor pair 552 drives a PWM output signal that is coupled to an lower external transistor that drives an external load such as transistor 302B shown in Fig. 3.

The output control logic 520 receives as inputs each of the configuration bits 505-515. The output control logic also receives a PWM signal for each PWM output signal pair 540 and 560. The output control logic also receives a PWM data signals 570 for outputting on each PWM output channel when the PWM module and individual PWM output channels are enabled. The PWM data signals 570 may be generated, for example, by the PWM generators 208A-D as shown in Fig. 2.

The PWM enable signals 565 are generally set to disable the PWM outputs during and after a power up of the processor 100 and also after a reset. They may also be disabled under program control and in other ways.

When the PWM module and PWM data channels are enabled by the PWM enable signal 565, the output control logic 520 is configured to drive each transistor pair 522 and 552 to place the PWM data from the PWM data signals 570 onto the respective PWM output pins 540 and 550.

When the PWM enable signals 565 indicate that PWM channels are disabled, then the output control logic stimulates the transistors within each transistor pair 522 and 552 to either place each output transistor pair in a tri-state mode, an active high mode or and active low mode. Considering the transistor pair 522, for example, in the tri-state mode, the output control logic drives both transistor 525 and 530 to the OFF state. In the active high mode, the output control logic 520 drives transistor 525 to the ON state and transistor 530 to the OFF state. In the active low mode, the output control logic 520 drives transistor 525 to the OFF state and transistor 530 to the ON state.

Whether each transistor pair 522 and 552 is place in a tri-state, active high or active low mode when the PWM enable signal is set to disable is determined by the configuration bits according to the description below.

The tri-state control bit determines whether output control logic causes the transistor pairs 522 and 552 to assume tri-state conditions when the corresponding PWM enable signal is set to disable (or not enabled). According to one embodiment of the invention, the tri-state control bit causes the output control logic 520 to place the device pairs into the tri-state condition when the tri-state control bit is in the un-programmed state. When the tri-state control bit is in the programmed state, the tri-state control bit causes the output control logic 520 configure the transistor pair 522 to assume either active high or active low states according to the high device

set bit 510 and causes the output control logic 520 to configure the transistor pair 552 to assume either active high or active low states according to the low device set bit 510.

According to one embodiment of the present invention, for example, when the tri-state control bit 505 is programmed, and the high device set bit is un-programmed, the output control logic drives the transistor pair 522 to produce an active high voltage level, i.e., the output 540 is driven to the power voltage. When the tri-state control bit 505 is programmed, and the high device set bit is programmed, the output control logic drives the transistor pair 522 to produce an active low voltage level, i.e., the output 540 is driven to the ground or low voltage level. The output 540 is provided to an external "high" device such as the transistor 302A.

According to another embodiment of the present invention, for example, when the tri-state control bit 505 is programmed, and the low device set bit is un-programmed, the output control logic drives the transistor pair 522 to produce an active high voltage level, i.e., the output 560 is driven to the power voltage. When the tri-state control bit 505 is programmed, and the low device set bit is programmed, the output control logic drives the transistor pair 522 to produce an active low voltage level, i.e., the output 560 is driven to the ground or low voltage level. The output 560 is provided to an external "low" device such as the transistor 302B.

In this manner, a user may program the configuration bits to match a particular system configuration. The configuration bits cause the PWM output signals to drive external devices such as the transistor pair comprising transistors 302A and 302B to known states that will not dissipate power or cause damage to the devices. In addition, the need for external resistors is avoided. Moreover, the configuration bits may be reprogrammed in non-volatile memory in some embodiments to permit flexible implementation within a system.

The control bits 510 and 515 also may be used to control the polarities of the PWM signals when the PWM module is enabled and operating. Setting the control bits in non-volatile memory provides a robust solution that is operable after power up and reset events and also avoids problems of mis-setting values in software which would may arise if volatile memory is used.

While specific embodiments of the present invention have been illustrated and described, it will be understood by those having ordinary skill in the art that changes may be made to those embodiments without departing from the spirit and scope of the invention. It will be understood, for example, that the assignment of functionality to programmed and unprogrammed states is arbitrary and may be changed accordingly.